ARL-TR-8285 ● JAN 2018

US Army Research Laboratory

Smoke Screen in Cyberspace Architecture Document

by Chien Hsieh and Andrew Toth

Approved for public release; distribution is unlimited.

NOTICES

Disclaimers

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.

ARL-TR-8285 ● JAN 2018

US Army Research Laboratory

Smoke Screen in Cyberspace Architecture Document

by Chien Hsieh
ICF International, Fairfax, VA

Andrew Toth
Computational and Information Sciences Directorate, ARL

Approved for public release; distribution is unlimited.

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 0704-0188*

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing the burden, to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.
PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.

1. REPORT DATE *(DD-MM-YYYY)*	**2. REPORT TYPE**	**3. DATES COVERED (From - To)**
January 2018	Technical Report	1 January–30 September 2017

4. TITLE AND SUBTITLE	
Smoke Screen in Cyberspace Architecture Document	**5a. CONTRACT NUMBER**
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	**5d. PROJECT NUMBER**
Chien Hsieh and Andrew Toth	**5e. TASK NUMBER**
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	**8. PERFORMING ORGANIZATION REPORT NUMBER**
US Army Research Laboratory ATTN: RDRL-CIN-T Aberdeen Proving Ground, MD 21005-5067	ARL-TR-8285
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	**10. SPONSOR/MONITOR'S ACRONYM(S)**
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Networks at the tactical edge must be resilient to cyber and electromagnetic operations by a capable adversary; even when partly compromised, they should remain opaque to the adversary and effective for friendly forces. The Smoke Screen in Cyberspace seedling project funded by the Assistant Secretary of Defense for Research and Engineering Science and Technology seeks to research methods to perform radical fragmentation (splitting) of friendly data into a large number of fragments (cyber “smoke”) and continually maneuver them across multiple devices of the edge network to gain resiliency to adversary electronic warfare, cyber, and kinetic attacks and intercept and enable agile maneuvering of data, rapid recovery, obfuscation and deception. This report documents the architecture of the US Army Research Laboratory–developed prototype application that enables researchers to conduct experiments in both emulation and wireless environments and produces metrics to analyze experiment results. This application provides a mechanism to successfully demonstrate the functions of data splitting, dispersion, and reassembly.

15. SUBJECT TERMS

data fragmentation, data dispersion, cyber, network resiliency, data obfuscation, data deception

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
					Chien Hsieh
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			**19b. TELEPHONE NUMBER (Include area code)**
Unclassified	Unclassified	Unclassified	UU	30	391-394-2365

Standard Form 298 (Rev. 8/98)
Prescribed by ANSI Std. Z39.18

Contents

Approved for public release; distribution is unlimited.

List of Figures

List of Tables

INTENTIONALLY LEFT BLANK.

1. Introduction

Smoke Screen in Cyberspace is an Assistant Secretary of Defense for Research and Engineering Science and Technology seedling project tasked to research methods to perform radical fragmentation (splitting) of friendly data into a large number of fragments (cyber "smoke") and continually maneuver them across multiple devices of the edge network to gain resiliency to adversary electronic warfare, cyber, and kinetic attacks and intercept and enable agile maneuvering of data, rapid recovery, obfuscation, and deception.

The main motivation for the project is to explore methods to make networks at the tactical edge resilient to cyber and electromagnetic operations by a capable adversary. Even when partly compromised, the networks should remain opaque to the adversary and effective for friendly forces.

Data splitting for security and scalability is practiced in many modern commercial data stores (e.g., Voldemort of LinkedIn[1,2], Dynamo of Amazon[3]), but not for edge devices. With the maturing of the edge network-distributed data store, Gaian Database (GaianDB),[4–6] a product of the US Army Research Laboratory's (ARL's) UK-US Network and Information Sciences International Technology Alliance (NIS-ITA),[7] we can explore the first tactical use of data splitting.

This report documents the work done during the first phase of the seedling project. The tasks in this phase seek to adapt and integrate GaianDB and some data distribution technologies such as those employed by Dynamo Database. The majority of the effort centered around building the Smoke Screen prototype application that enables researchers to conduct experiments in both emulation and wireless environments and produces metrics to analyze experiment results. Specifically, this application provides a mechanism to successfully demonstrate the functions of data splitting, dispersion, and reassembly.

The functionality of management of the fragments and maneuvering of data fragments around the tactical network is not yet implemented. Those are objectives for later phases of the Smoke Screen project.

2. Requirements

2.1 Data Fragmentation and Distribution

For this requirement, the Smoke Screen prototype is to implement necessary logic that adequately splits up data into fragments and stores them in devices. Each data fragment must be unrecognizable in terms of not yielding clues as to the type, format, and content of the original data. In addition, the distribution of the data fragments among devices must be randomized in order to not demonstrate any pattern in data communication.

2.2 Data Reassembly

The prototype is also to implement a mechanism where data fragments are collected and reassembled into the original form of the data.

2.3 Easily Adaptable to Emulation and Physical Environments

The implementation of the prototype must be easily staged in the CORE, EMANE, and physical environments, where researchers can conduct experiments of basic Smoke Screen operations based on designed tactical scenarios.

3. Technology Research

3.1 GaianDB

The Gaian Database, or GaianDB, is a Dynamic Distributed Federated Database developed by IBM UK's Strategy and Technology Software Group. It is constructed on top of the Apache Derby[8] database engine. It is written in Java and has a small footprint of less than 4 MB, which makes it suitable to run on a wide variety of devices.

Primary attributes of GaianDB are its ability to automatically discover other GaianDB nodes in the same operating environment, and for each node to federate multiple data sources, such as comma-separated files, Structured Query Language (SQL)-compliant databases, and Microsoft Excel files. GaianDB uses a biologically inspired connectivity model to efficiently organize the discovered nodes in a way that minimizes the network diameter while it maximizes connections to most fit nodes. Furthermore, inherent in GaianDB is the flexible "store locally, query anywhere" (SLQA) pattern, which means data can be located in its source locations while allowing queries to be injected from anywhere in the GaianDB network.

Another powerful feature GaianDB provides is distributed subqueries. This feature automatically propagates a query that was introduced at one of the GaianDB nodes to all of the other GaianDB nodes in the network. Once the subquery arrives at a node, it will run locally at that node, with the results fed back to the original querying node and aggregated into the final result set. The subqueries are not limited to "select" statements. They can contain insert/update/delete statements as well, so long as they run against physical resources such as Derby tables. Another important aspect of the query propagation algorithm is that redundant queries are discarded, so a node will not execute the same query multiple times.[5]

All these characteristics make GaianDB an excellent candidate to act as the data transport and persistence engine for the Smoke Screen prototype. For the Smoke Screen in Cyberspace program, a GaianDB node resides in each of the devices in the tactical network, and it is through the GaianDB node and the associated GaianDB network the devices can maneuver the data.

3.2 Dynamo Database and Consistent Hashing

Dynamo database is a NoSQL solution that Amazon deployed as its highly available key-value storage system to provide the "always-on" experience to its customers. Faced with reliability requirements of massive proportion, Amazon needed a platform to provide services on top of an infrastructure consisting of thousands of servers and components spanning many data centers across the globe. Amazon's Dynamo achieved scalability and availability by applying effective techniques to address the problem of data partitioning, highly available writes, temporary failure handling, and recovery of permanent failures. The success story of the Dynamo solution has inspired other solutions such as Project Voldemort, Apache Cassandra,[9] Riak, and so on.

Notable in the Dynamo implementation is its use of the Consistent Hashing[10] technique to achieve effective and scalable data partition and replication. As per the Wikipedia description, this hashing is characterized by the fact that when a hash table is resized, only K out of n keys need to be remapped on average, where K is the number of keys and n is the number of slots. Most importantly, this concept involves pseudo-randomly distributed points and minimizes impact on value mapping when storage nodes are removed or added. The Consistent Hashing method can be applied to Smoke Screen to randomize distribution of data fragments.

Figure 1 depicts the system architecture of Smoke Screen in which 3 Smoke Screen nodes are active in the environment. The following sections describe the major components of the architecture.

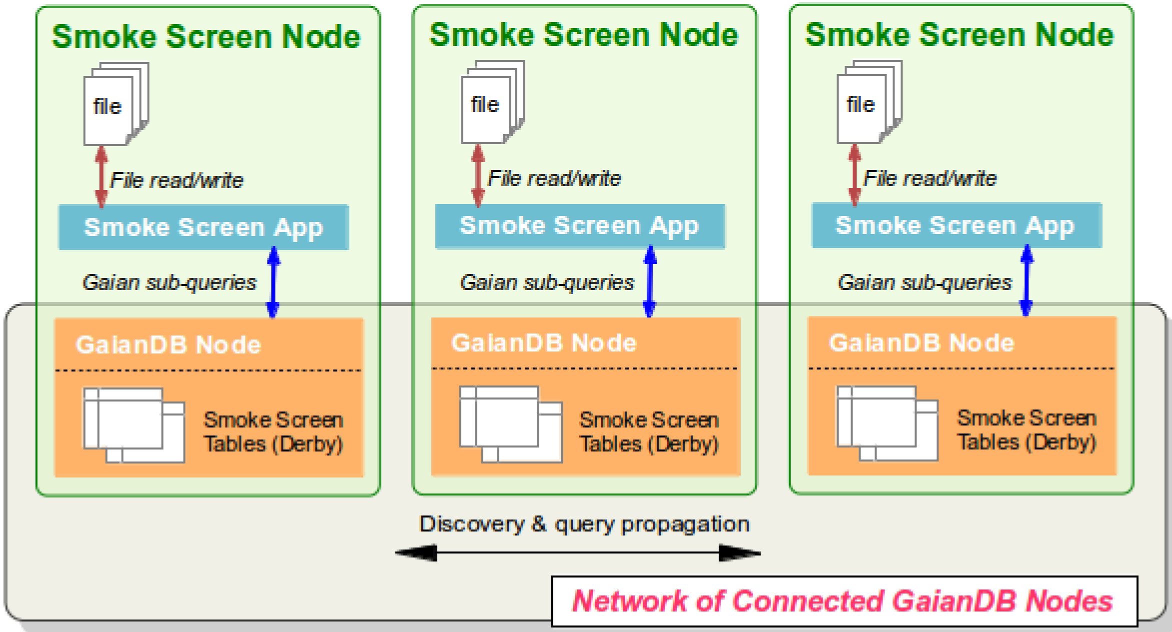

Fig. 1 Smoke Screen architecture

4. Architecture

4.1 GaianDB Network

This is the network of connected GaianDB nodes. These nodes and the associated network provide the communication backbone and data storage infrastructure for Smoke Screen. Each device in the tactical network will hold a GaianDB node.

The GaianDB network can be viewed as an autonomous infrastructure where GaianDB nodes are able to discover each other, as well as to execute and propagate injected queries. Wrapped inside each GaianDB node is a set of Smoke Screen-related database tables. They are physical Derby tables designed to store data fragments (cyber smoke), their meta-information, and any relevant attributes of the GaianDB node.

4.2 Smoke Screen Node

A Smoke Screen node is a logical unit that is formed when a Smoke Screen operation (e.g., data distribution or data reassembly), is being executed. A Smoke Screen operation is to be initiated by a user on a device in the tactical network.

During the execution of the Smoke Screen operation, the Smoke Screen application has access to the local file system on the host device in order to read or write files. In addition, it establishes connection to the GaianDB node on the device, through which it can issue appropriate distributed subqueries (select/insert statements) to the GaianDB network to fulfill the specified Smoke Screen operation.

4.3 Smoke Screen Application

The Smoke Screen application is a Java-based prototype application developed by ARL researchers for the initial phase of the Smoke Screen seedling project. This application contains the logic and implementation of algorithms to support Smoke Screen operations of data fragmentation and distribution and data reassembly. This application is invoked from the command line with appropriate execution options.

5. Database Design

GaianDB is used extensively by Smoke Screen as a provider for data persistence and manipulation. Derby is the underlying database mechanism used for GaianDB.

5.1 Derby Tables

Table 1 describes the Derby database tables and their respective columns that are required in each GaianDB node in order to support Smoke Screen operations.

Table 1 Derby Database Tables

Table Name	Columns
node_info	`node_id: varchar(128)  -- MD5 hash of node`

data_fragment	`node: varchar(128)  -- name of origination node of data`
	`resource: varchar(128)  -- name of the file`
	`seq: varchar(128)  -- sequence number of the data fragment`
	`data: varchar(32000)  -- the actual data fragment`
	`hash: varchar(128)  -- hash of the record for integrity check`

5.2 GaianDB Distributed Subqueries

The feature of Gaian distributed subqueries is integral to the operations of Smoke Screen. The Smoke Screen application issues a wide variety of distributed subqueries, which fall into one of the following categories:

- **GaianDB nodes and environment.** These subqueries provide information about local and discovered remote GaianDB nodes that are capable of participating in Smoke Screen operations.
- **Data distribution.** These subqueries specify the content and destination of data fragment insertion to GaianDB nodes.
- **Data fragment queries.** These subqueries allow Smoke Screen to determine available data fragments in the GaianDB nodes, as well as to retrieve the actual data fragments.

6. Functional Design

6.1 Initialization

When the Smoke Screen application starts up, it undergoes the following set of operations:

- **Establish a database connection to GaianDB.** Using information provided by the user, the Smoke Screen application confirms that it is able to communicate with the local GaianDB node.
- **Identify all peer GaianDB nodes.** The Smoke Screen application determines the names and associated information about all the GaianDB nodes in the network. This information is required for any Smoke Screen action requested by the user.

Approved for public release; distribution is unlimited.

6.2 Data Fragmentation

Before friendly data can be distributed and stored in GaianDB nodes, Smoke Screen breaks up the data into fragments, or "cyber smokes". The data fragmentation process, illustrated in Fig. 2, consists of 2 parts: encryption and partitioning. This process is similar to the technique proposed by Deswarte et al.[11]

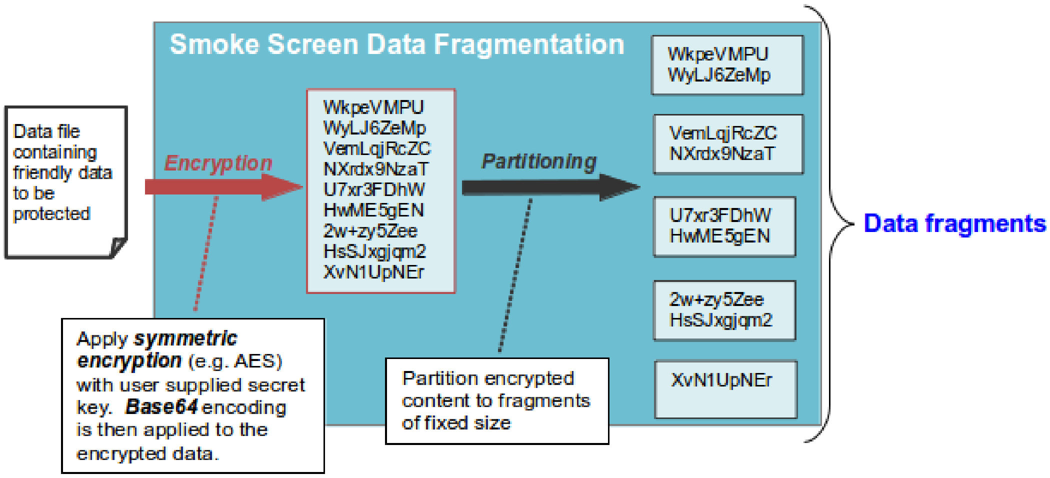

Fig. 2 Smoke Screen data fragmentation processing

6.2.1 Encryption

In order to maximize the effect of data obfuscation, encryption is applied to the data before it is partitioned. Advanced Encryption Standard (AES) is used as the encryption algorithm. AES is a standard symmetric-key algorithm and has been adopted for use by the US Government. After data are encrypted using a user-supplied secret key string, the resulting encrypted content undergoes Base64 encoding, which converts binary data into ASCII string format. This step is necessary because the output of AES encryption is in binary format, and the distributed subquery interface of GaianDB only handles data in ASCII string format.

6.2.2 Partitioning

The ASCII-formatted encrypted data are partitioned into data fragments of equal size. The data fragment size is configurable by the user. The Smoke Screen application caps it at 30,000 bytes to ensure the fragments are of reasonable size.

Each fragment comes across as random sequences of characters, and the only way to reconstitute the file would be to assemble the fragments in the correct order, apply Base64 decoding to obtain the binary encrypted content, and then decrypt the content to obtain its original form.

6.3 Data Distribution

The Smoke Screen application uses the Consistent Hashing algorithm to determine the storage locations (GaianDB nodes) for the data fragments. In addition, if specified by the configuration, a data fragment can be stored in multiple locations for redundancy.

6.3.1 Consistent Hashing

As mentioned previously, Amazon's Dynamo uses the Consistent Hashing algorithm to satisfy its data partitioning and replication requirements.[12] Smoke Screen adopted a similar algorithm, as proposed by Tom White in his Consistent Hashing article,[13] to determine storage locations for the data fragments. This section summarizes the approach.

The algorithm uses the concept of an integer number ring. It does so by mapping the range of integer values, from -2^{31} to $2^{31} - 1$, into a ring so the values wrap around. During a Smoke Screen operation, the algorithm maps the storage nodes (e.g., the discovered GaianDB nodes) into the number ring. This is accomplished by taking the MD5 hash values of the node names and converting them into integer values. When it is time to determine the storage location of a given data fragment, the algorithm takes the integer representation of the MD5 hash value of the data fragment, and then it moves clockwise around the ring until the storage node is found. For example, in Fig. 3, fragments 1 and 3 are stored in node C, fragment 2 in node A, and fragment 4 in node B.

Approved for public release; distribution is unlimited.

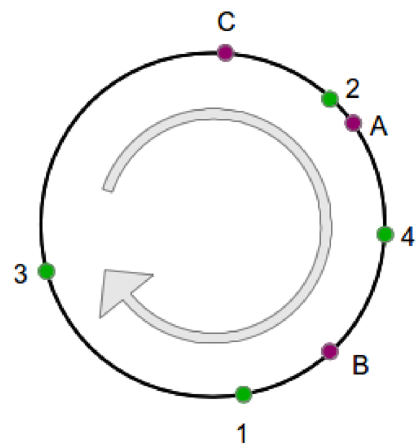

Fig. 3 Consistent Hashing concept

As can be seen in Fig. 3, the separation intervals between the storage nodes on the ring are not always, or likely, to be even, therefore resulting in nonuniform distribution of fragments among the storage nodes. To mitigate the problem, the concept of “virtual nodes” is introduced, where additional representations of the storage nodes are inserted to the ring. These virtual nodes are generated by placing additional random hash values on the ring and associating them with the actual storage nodes, thereby increasing the likelihood of randomized data distribution.

6.3.2 Data Redundancy

In order to overcome the issue of compromised or lost devices, a fragment can be stored at multiple nodes, essentially creating replicas, to provide redundancy. An additional storage node is obtained by first appending a unique string, consisting of the current time and a randomly generated number, to the data fragment, and then running this altered form of data fragment through the consistent hashing algorithm to identify the storage node. This process is repeated for the number of desired redundant storage nodes. The data redundancy logic tracks the set of storage nodes and ensures all storage nodes are distinct.

Storing multiple fragment replicas does not require issuing multiple Gaian distributed subqueries. Rather, the additional storage node information is simply embedded in the same subquery.

6.3.3 Fragment Storage

Once the storage GaianDB nodes for a data fragment are determined, the Smoke Screen application generates the appropriate distributed subqueries so the data fragment can be inserted to the targeted nodes. As stated previously, the subqueries are propagated to all nodes in the GaianDB network. Therefore it is only injected to the GaianDB node local to the Smoke Screen application, and it will be automatically propagated to all the GaianDB nodes in the network. The subqueries are constructed so that each contains the data fragment content, plus conditional clauses that indicate whether the fragment is to be stored at a particular GaianDB node.

The following example SQL statement illustrates the format and structure of a typical GaianDB distributed subquery that inserts a data fragment.

```
select gdb_node, update_count from new com.ibm.db2j.GaianQuery('
  insert into data_fragment (node, resource, seq, data, hash)
  values(''dev1'',       -- node id of the data file origin
         ''file_1'',     -- file name
         ''3'',          -- sequence number of the data fragment
         ''61N0JACWybW+E4AhezOpdjKaOnTTEM3AU89IvtKW7UKfy6WymsBXOvYt'', --
data fragment
         ''1559ec74af108543aeb300a84d14172d'') -- used as checksum
  ', 'with_provenance') Q
where gdb_node='dev3'  -- the node ID where fragment is to be stored
   or gdb_node='dev6'  -- another node ID where fragment is to be stored
```

The data distribution process is summarized in Fig. 4.

Approved for public release; distribution is unlimited.

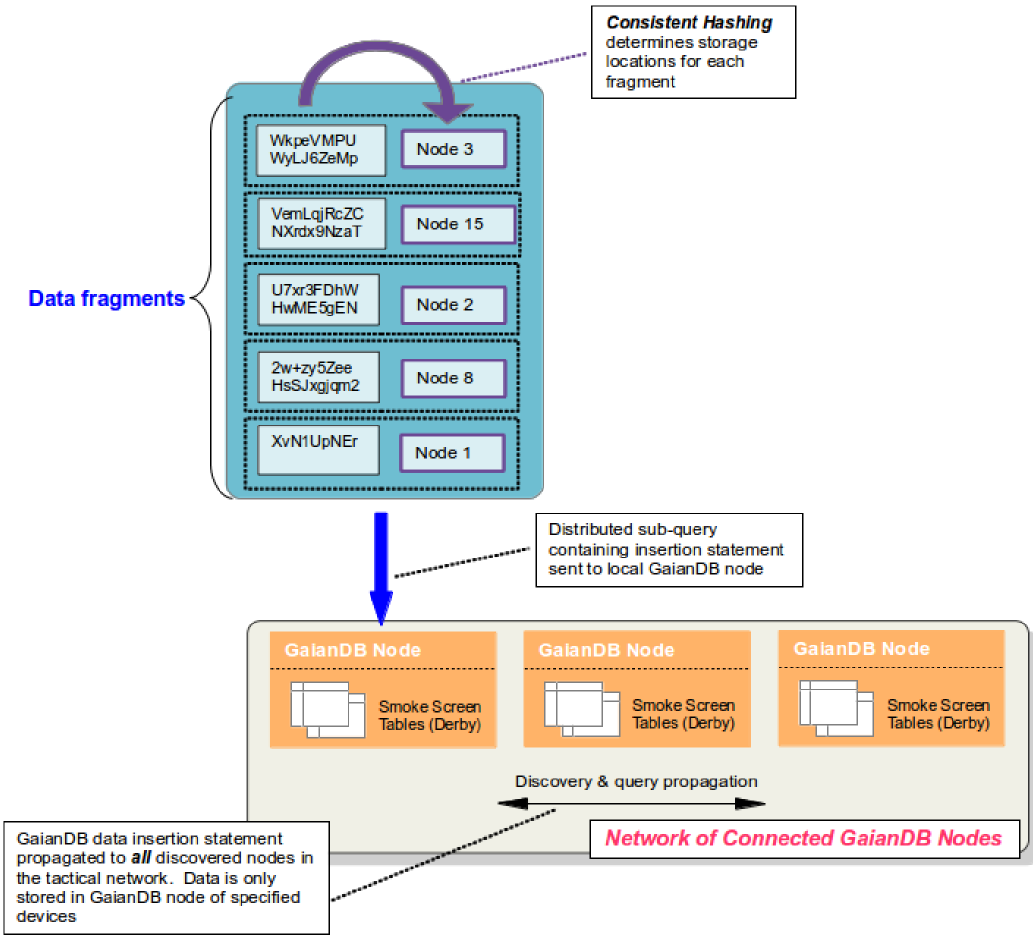

Fig. 4 Smoke Screen data fragment distribution processing

6.4 Data Reassembly

The data reassembly is the reverse process of the data fragmentation and distribution. Because Smoke Screen does not include a data management layer, there is no centralized view or control of the data fragments. In other words, there is no definitive information about the total number of data fragments of a file, or that of GaianDB nodes holding any particular data fragment.

Therefore, the data reassembly mechanism employs an iterative process. It requests and acquires data fragments from one GaianDB node at a time. When data fragments pertaining to a specific file are retrieved from a particular node, the Smoke Screen application joins all accumulated data fragments in the correct sequence and attempts to obtain the original data by decrypting the assembled content. The decryption process will fail if there is any missing data fragment. In such a case, Smoke Screen will proceed to request data from the next available GaianDB node. This process continues until all data fragments are collected and the original content is decrypted successfully from the assembled content. It is possible for the reassembly process to fail if it is unable to collect all data fragments related to a file.

The following sections discuss additional details of the reassembly process.

6.4.1 Fragment Retrieval

When GaianDB nodes are discovered and federated, each node tracks the “depth” of all the other nodes in the network, which represents the number of hops it would take for a query to reach from the local node to a peer node. The depth of the local node is zero. The higher the depth value, the longer it takes for a query to reach.

In order for the data collection process to be more efficient, it always starts with the local GaianDB node. If the data reassembly process determines that not all data fragments have been accumulated, it proceeds to request for data fragments from other GaianDB nodes in the order of their respective depth values.

The following example SQL statement illustrates the format and structure of a typical GaianDB distributed subquery that requests data fragments of a particular sequence identification.

```
select gdb_node, node, resource, seq, data, hash from new
com.ibm.db2j.GaianQuery('
  select node node, resource resource, seq seq, data data, hash hash from
data_fragment
  where node=''dev1''         -- node id of the data file origin
    and resource=''file_1''   -- file name
    and seq=''3'',            -- sequence number of the data fragment
  ', 'with_provenance') Q
where gdb_node='dev4'        -- the node ID where data fragment is stored
```

6.4.2 Decryption

The decryption processing first performs Base64 decoding of the assembled fragments to obtain the encrypted content in binary form. Then it applies the AES decryption function on the encrypted binary data using the secret key that was used to encrypt the data in the first place.

Figure 5 summarizes the data reassembly process.

Approved for public release; distribution is unlimited.

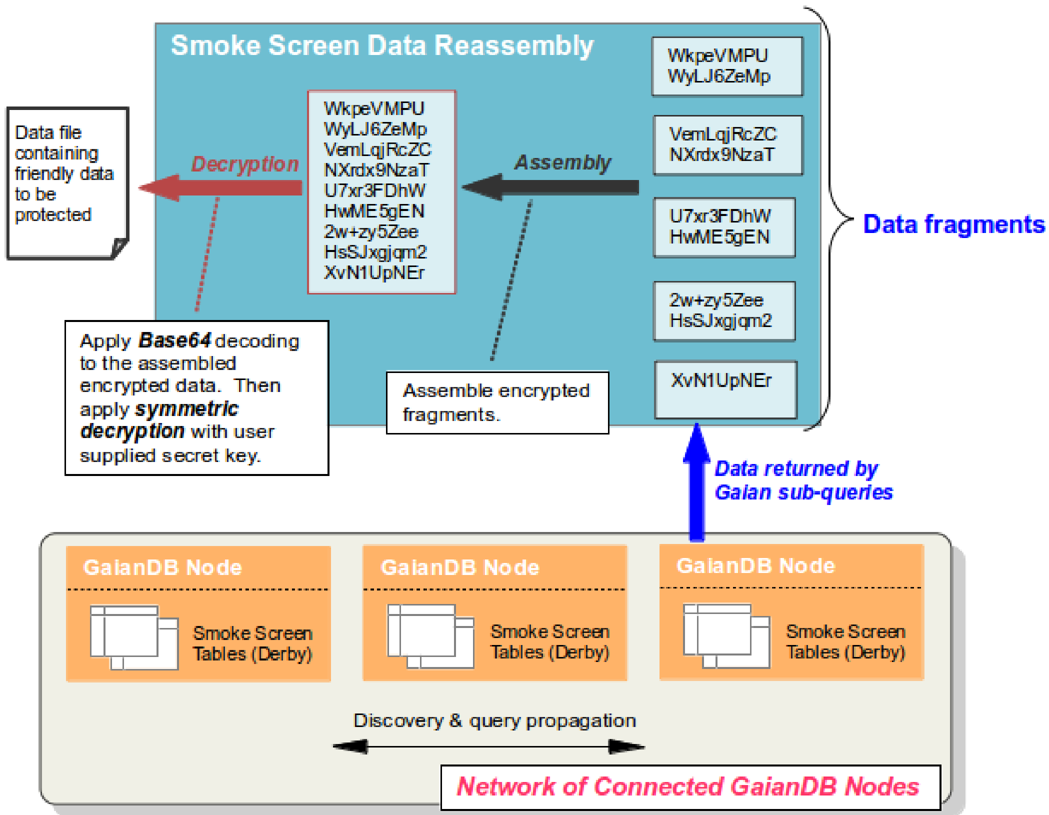

Fig. 5 Smoke Screen data reassembly processing

6.4.3 Impact of Compromised or Lost Devices during Data Assembly

The assembly of data will be successful so long as the number of lost devices (L) is less than the number of replicas (R) for the data fragments, L is less than R. This assumes every data fragment of the file has the same number of replicas.

7. Adaptation for Emulation Environments

After the Smoke Screen prototype was functionally tested in wired networks, the next logical step was to validate it in emulation environments. Emulation facilitates a methodical approach to function validation while allowing greater variability and control of the test environment.

7.1 Emulation Environment

The Extended Mobile Ad-hoc Network Emulator (EMANE) is an open source project developed by AdjacentLink initially sponsored by the US Naval Research Laboratory (NRL) and in cooperation with ARL.[14] It is a framework that provides a highly extensible wireless protocol and application test environment. In addition, EMANE offers physical layer models to account for signal propagation, antenna profile effects, and interference sources to provide a realistic environment for wireless network stack development.

Running the Smoke Screen application in EMANE will facilitate the demonstration and verification of the basic functions of data fragmentation, data distribution, and data reassembly under a variety of tactical scenarios. Because the Smoke Screen application is written in Java, it can be invoked directly in EMANE as long as the Java Runtime Environment is installed in EMANE. One additional module that was developed for running Smoke Screen in EMANE was a command interpreter module. Through a user-defined command script file, this module provides a mechanism for orchestrating sequences of Smoke Screen commands that can be executed in accordance with the requirements of any particular tactical scenario. The command script file is in JSON format, and each command in the file can be used to instruct a particular Smoke Screen node to distribute or reassemble a file at any given time.

Figure 6 contains the architecture of Smoke Screen when deployed in EMANE.

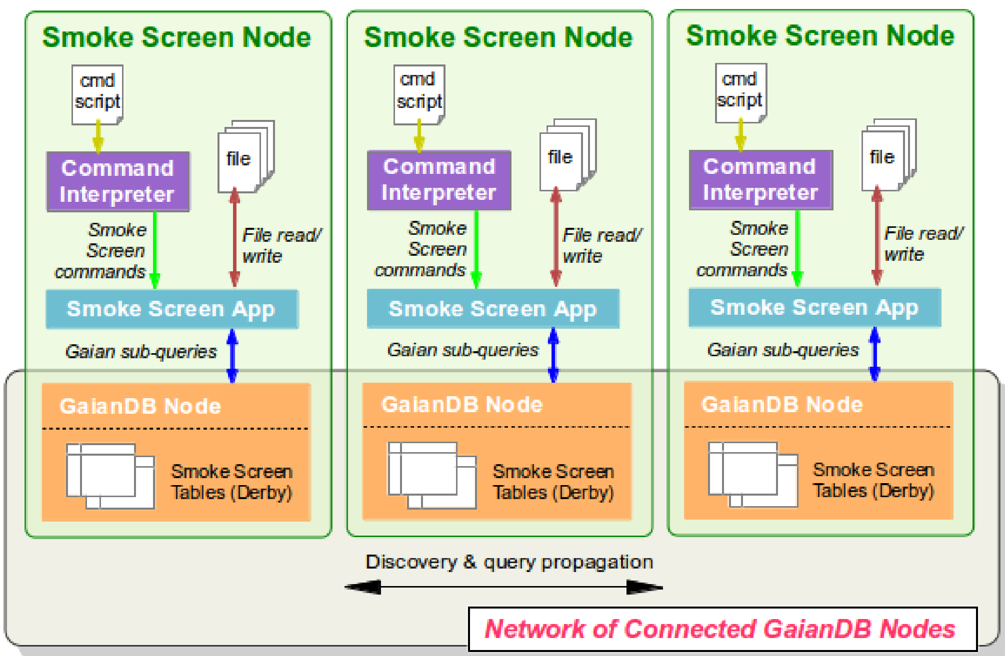

Fig. 6 Smoke Screen in EMANE

Approved for public release; distribution is unlimited.

7.2 Input Parameters

The following is a summary of variables used to conduct Smoke Screen functional tests in EMANE:

- **File.** A JPEG file of size 42,882 bytes was used in all file distribution tests. The same file was used for verification of file reassembly tests.
- **Number of nodes.** Tests were run with 2, 3, 5, and 10 Smoke Screen nodes in the emulation environment.
- **Data fragment size.** Settings for data fragment size were 500; 1,000; 2,000; 3,500; 5,000; 10,000; and 20,000 bytes.
- **Data rate.** Data rates of 50 kbps, 100 kbps, 250 kbps, 500 kbps, and 1 Mbps were tested.
- **Radio model.** RF-PIPE was used in all tests.
- **Node distances.** The distances between the nodes varied from 0.1 to 0.6 miles. The nodes were static.

7.3 Tests and Metrics

The Smoke Screen features of file distribution and file reassembly were tested in EMANE. Time to completion was the main metric collected in the experiments. Figures 7–9 illustrate outputs from selected sets of data. In general, time to completion of data distribution degrades significantly when data fragment size drops below 2,000 bytes. On the other hand, when data fragment size and network bandwidth are constant, the time to completion generally increases as the number of nodes increases.

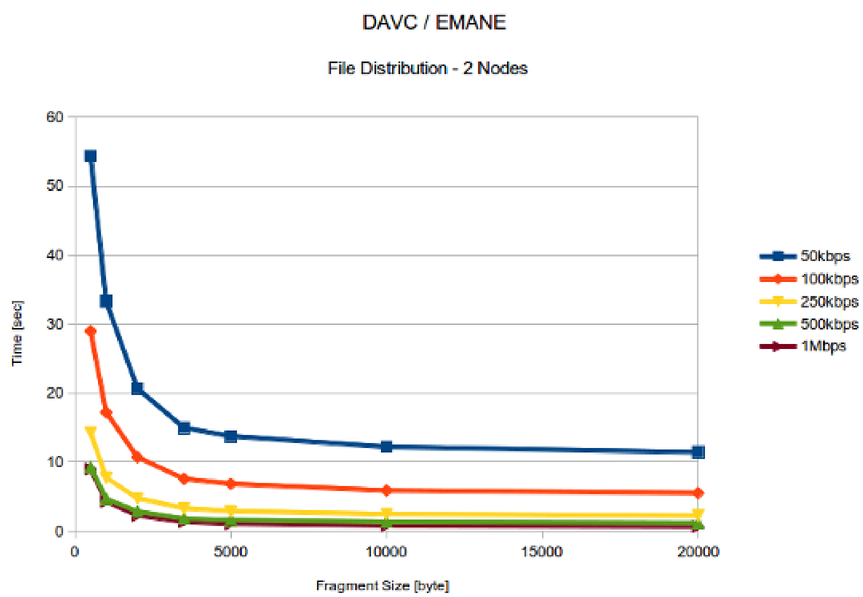

Fig. 7 File distribution results in EMANE: 2 nodes

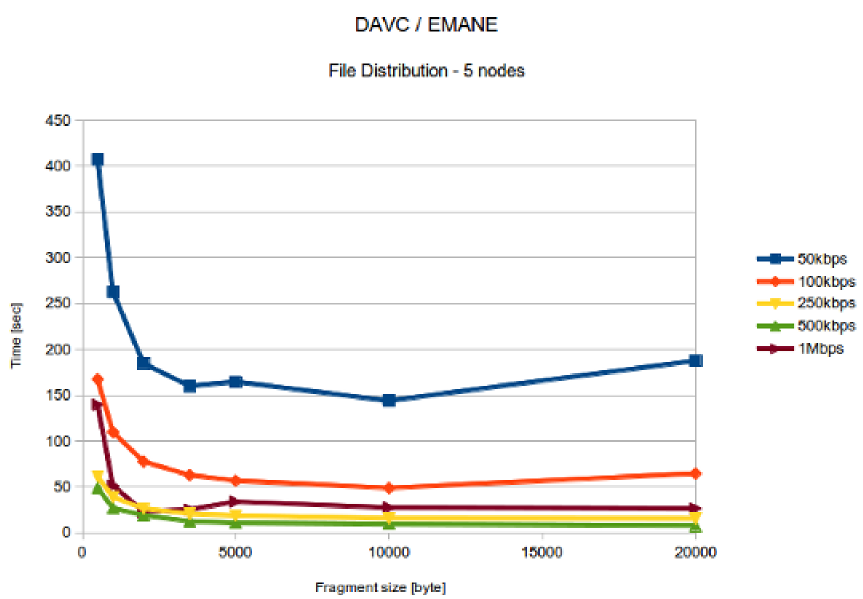

Fig. 8 File distribution results in EMANE: 5 nodes

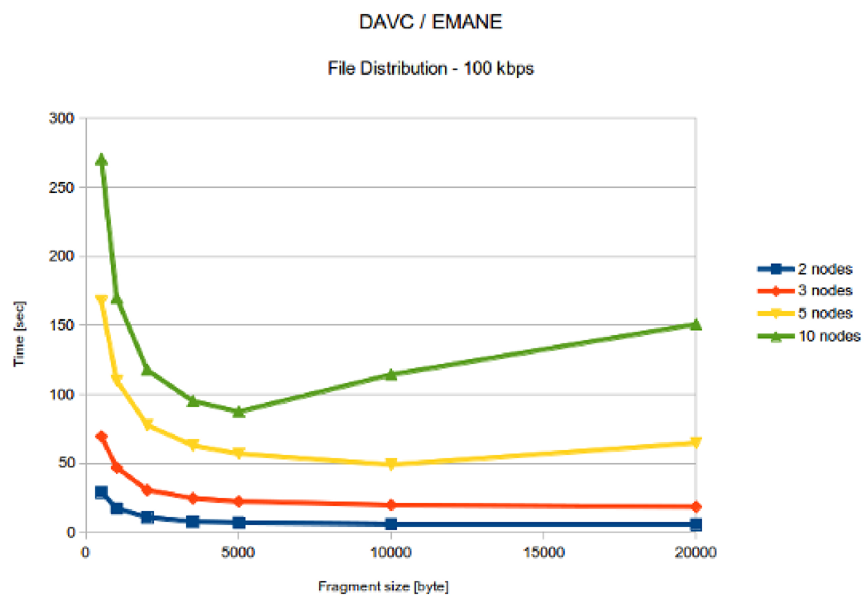

Fig. 9 File distribution results in EMANE: 100 kbps

8. Conclusion

The Smoke Screen application outlined in this report successfully demonstrates the basic operations of data fragmentation, distribution, and reassembly in tactical networks. It provides the foundation for future development for the Smoke Screen project. The following are functions to be implemented in future phases of the project:

- **Alternative data distribution mechanism.** Emulation test data showed that data distribution efficiency degraded as number of nodes increased. Therefore, it will be worthwhile to explore other data routing strategies to improve the scalability of Smoke Screen. Another concept to explore would be clustering or hierarchy of nodes, where the responsibility of data distribution can be divided and shared.
- **Data management layer.** This layer will provide comprehensive control of data storage in terms of what and where data are located. Essentially this layer provides complete knowledge of all resources stored in a given Smoke Screen environment.
- **Continual data maneuvering.** In order to increase resiliency to threats and attacks, data are to be maneuvered around the tactical network on a regular basis. This should be done autonomously by the Smoke Screen engine.

9. References

1. Project Voldemort. [accessed 2018 Jan 10]. http://www.project-voldemort.com/voldemort/.

2. Design of Voldemort. [accessed 2018 Jan 10]. http://www.project-voldemort.com/voldemort/design.html.

3. Wikipedia. Dynamo (storage system). Wikipedia; 2017 Oct 13 [accessed 2018 Jan 10]. https://en.wikipedia.org/ wiki/Dynamo_(storage_system).

4. IBM Gaian database [accessed 2018 Jan 10]. https://www.ibm.com/developerworks/community/groups/service/html /communityview?communityUuid=f6ce657b-f385-43b2-8350-458e6e4a344f.

5. Brady J. Gaian Database technology seeks to optimize data identification and processing. Aberdeen Proving Ground (MD): Army Research Laboratory (US); 2013 Dec 19 [accessed 2018 Jan 10]. https://www.army.mil/article/117092 /Gaian_Database_technology_seeks_to_optimize_data _identification_and_processing.

6. Bent G, Dantressangle P, Vyvyan D, Mowshowitz A, Mitsou V. A dynamic distributed federated database. ResearchGate [accessed 2018 Jan 10]. https://www.researchgate.net/profile/Patrick_Dantressangle /publication/262674503_A_Dynamic_Distributed_Federated_Database/links/ 00b7d5386119b2c4ff000000.pdf.

7. International Technology Alliance in Network and Information Sciences [accessed 2018 Jan 10]. http://nis-ita.org.

8. Apache Derby. Apache Software Foundation [accessed 2018 Jan 18]. https://db.apache.org/derby/.

9. Apache Cassandra. Apache Software Foundation [accessed 2018 Jan 10]. http://cassandra.apache.org.

10. Consistent hashing. Wikipedia; 2017 Sep 30 [accessed 2018 Jan 10]. https://en.wikipedia.org/wiki/Consistent_hashing.

11. Deswarte Y, Blain L, Fabre J-C. Intrusion tolerance in distributed computing systems. IEEE Symposium on Research in Security and Privacy; 1991 May 20–22; Oakland, CA.

12. DynamoDB replication and partitioning part 4 [accessed 2018 Jan 10]. http://cloudacademy.com/blog/dynamodb-replication-and-partitioning-part-4/.

13. White T. Consistent hashing. 2007 Nov 27 [accessed 2018 Jan 10]. http://www.tom-e-white.com/2007/11/consistent-hashing.html.

14. US Naval Research Laboratory Networks and Communications Systems Branch. Extendable Mobile Ad-hoc Network Emulator (EMANE). Washington (DC): Naval Research Laboratory (US) [accessed 2018 Jan 10]. https://www.nrl.navy.mil/itd/ncs/products/emane.

List of Symbols, Abbreviations, and Acronyms

AES	advanced encryption standard
ARL	US Army Research Laboratory
EMANE	Extended Mobile Ad-hoc Network Emulator
GaianDB	Gaian Database
NIS-ITA	Network and Information Sciences International Technology Alliance
NRL	US Naval Research Laboratory
SLQA	store locally, query anywhere
SQL	Structured Query Language

Glossary

Smoke Screen in Cyberspace	Also known as Smoke Screen. It is a seedling project funded by the Assistant Secretary of Defense for Research and Engineering Science and Technology that seeks to perform radical fragmentation and dispersion of friendly data into a large number of fragments in tactical networks, in order to provide resiliency against cyber threats and attacks of adversaries.
Derby Database	An Apache open source relational database implemented entirely in Java.
Gaian Database	A lightweight dynamically distributed federated database engine based on Apache Derby 10.
Smoke Screen Application	A Java-based prototype application developed during the first phase of the Smoke Screen project. This prototype demonstrated the basic functions of fragmentation, distribution and subsequent reassembly of friendly data.

1 (PDF) DEFENSE TECHNICAL
INFORMATION CTR
DTIC OCA

2 (PDF) DIR ARL
IMAL HRA
RECORDS MGMT
RDRL DCL
TECH LIB

1 (PDF) GOVT PRINTG OFC
A MALHOTRA

2 (PDF) ARL
RDRL CIN T
C HSIEH
A TOTH

Approved for public release; distribution is unlimited.